Lab Manual

for

Linear Algebra

by

Jim Hefferon

Cover: my Chocolate Lab, Suzy.

Preface

WARNING! This is an incomplete draft, no doubt riddled with errors.

This collection supplements the text *Linear Algebra*[1] with explorations to help students solidify and extend their understanding of the subject, using the mathematical software Sage.[2]

A major goal of any undergraduate Mathematics program is to move students toward a higher-level, more abstract, grasp of the subject. For instance, Calculus classes work on elaborate computations while later courses spend more effort on concepts and proofs, working less on the details of calculations.

The text *Linear Algebra* fits into this development process. Naturally it presents the material using examples and practice problems that are small-sized and have manageable numbers: an assignment to by-hand multiply a pair of three by three matrices of small integers will build intuition, whereas asking students to do that same question with twenty by twenty matrices of ten decimal place numbers would be badgering.

However, an instructor can be concerned that this misses a chance to make the point that Linear Algebra is widely applied, or to develop students's understanding through explorations that are not hindered by the mechanics of paper and pencil. Mathematical software can mitigate these concerns by extending the reach of what is reasonable to bigger systems, harder numbers, and longer computations. For instance, an advantage of learning how to handle larger jobs is that they are more like the ones that appear when students apply Linear Algebra to other areas. Another advantage is that students see new ideas such as runtime growth measures.

Well then, why not teach straight from the computer?

Our goal is to develop a higher-level understanding of the material so we want to keep the focus on vector spaces and linear maps. This exposition has the computations as a way to develop that understanding, not as the main point.

Some instructors may find that their students are best served by keeping a tight focus on the core material, and leaving aside altogether the work in this manual. Other instructors have students who will benefit from the increased reach that the software provides. This manual's existence, and status as a separate book, gives teachers the freedom to make the choice that suits their class.

[1] The text's home page http://joshua.smcvt.edu/linearalgebra has the PDF, the ancillary materials, and the LaTeX source.
[2] See http://www.sagemath.org for the software and documentation.

Why Sage?

In *Open Source Mathematical Software*[Joyner and Stein, 2007][1] the authors argue that for Mathematics the best way forward is to use software that is Open Source.

> Suppose Jane is a well-known mathematician who announces she has proved a theorem. We probably will believe her, but she knows that she will be required to produce a proof if requested. However, suppose now Jane says a theorem is true based partly on the results of software. The closest we can reasonably hope to get to a rigorous proof (without new ideas) is the open inspection and ability to use all the computer code on which the result depends. If the program is proprietary, this is not possible. We have every right to be distrustful, not only due to a vague distrust of computers but because even the best programmers regularly make mistakes.
>
> If one reads the proof of Jane's theorem in hopes of extending her ideas or applying them in a new context, it is limiting to not have access to the inner workings of the software on which Jane's result builds.

Professionals choose their tools by balancing many factors but this argument is persuasive. We use Sage because it is very capable so students can learn a great deal from it, and because it is Free[2] and Open Source.[3]

This manual

This is Free. Get the latest version from http://joshua.smcvt.edu/linearalgebra. Also see that page for the license details and for the LaTeX source. I am glad to get feedback, especially from instructors who have class-tested the material. My contact information is on the same page.

The computer output included here was generated automatically (I have automatically edited some lines). This is my Sage.

```
'Sage Version 5.2, Release Date: 2012-07-25'
```

Acknowledgements

I am glad for this chance to thank the Sage Development Team. In particular, without [Sage Development Team, 2012b] this work would not have happened. I am glad also for the chance to mention [Beezer, 2011] as an inspiration.

We emphasize practice.
–Shunryu Suzuki [2006]

[A]n orderly presentation is not necessarily bad but by itself may be insufficient.
–Ron Brandt [1998]

Jim Hefferon
Mathematics, Saint Michael's College
Colchester, Vermont USA
2012-Sep-10

[1]See http://www.ams.org/notices/200710/tx071001279p.pdf for the full text. [2]The Free Software Foundation page http://www.gnu.org/philosophy/free-sw.html gives background and a definition. [3]See http://opensource.org/osd.html for a definition.

Contents

Singular Value Decomposition

Recall that a line through the origin in $\mathbb{R}^n$ is $\{r\cdot\vec{v} \mid r\in\mathbb{R}\}$. One of the defining properties of a linear map is that $h(r\cdot\vec{v}) = r\cdot h(\vec{v})$. So the action of h on any line through the origin is determined by the action of h on any nonzero vector in that line.

For instance consider the line $y=2x$ in the plane.

$$\{r\cdot\begin{pmatrix}1\\2\end{pmatrix} \mid r\in\mathbb{R}\}$$

If $t\colon\mathbb{R}^2\to\mathbb{R}^2$ is represented by the matrix

$$\text{Rep}_{E_2,E_2}(t) = \begin{pmatrix}1&2\\3&4\end{pmatrix}$$

then here is the effect of t on one vector in the line.

$$\begin{pmatrix}1\\2\end{pmatrix}\overset{t}{\longmapsto}\begin{pmatrix}7\\10\end{pmatrix}$$

And here is the effect of t on other members of the line.

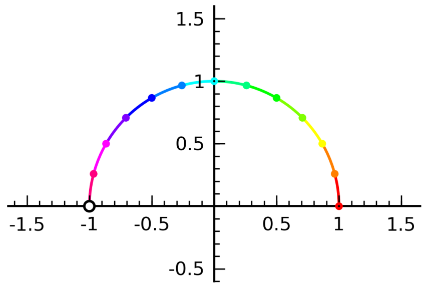
$$\begin{pmatrix}2\\4\end{pmatrix}\overset{t}{\longmapsto}\begin{pmatrix}14\\20\end{pmatrix}\qquad\begin{pmatrix}-3\\-6\end{pmatrix}\overset{t}{\longmapsto}\begin{pmatrix}-21\\-30\end{pmatrix}\qquad\begin{pmatrix}r\\2r\end{pmatrix}\overset{t}{\longmapsto}\begin{pmatrix}7r\\10r\end{pmatrix}$$

The map t's action on the line is uniform.

Unit circle

The above means that one way to describe a transformation's action is to pick one nonzero element from each line through the origin and describe where the transformation maps those elements. An easy way to select one nonzero element from each line through the origin is to take the upper half unit circle.

$$U=\{\begin{pmatrix}x\\y\end{pmatrix} \mid x=\cos(t),\ y=\sin(t),\ 0\leqslant t<\pi\}$$

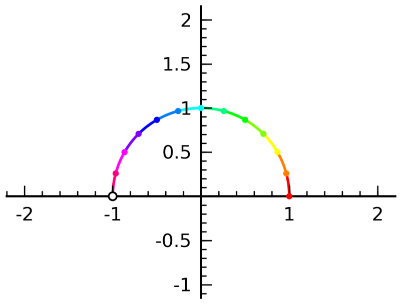


We produce the above graph by using a routine that draws the effect of an arbitrary matrix on the unit circle, and giving that routine the identity matrix. The colors will keep track the directionality in the set of before and after pictures below.

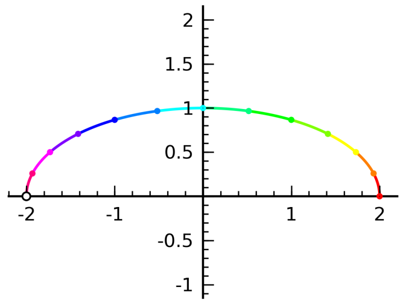
```
sage: load "plot_action.sage"
sage: p = plot_circle_action(1,0,0,1)   # identity matrix
sage: p.set_axes_range(-1.5, 1.5, -0.5, 1.5)
sage: p.save("sageoutput/plot_action10.png")
```

Here is the transformation doubles the x components of all vectors.

```
sage: p = plot_circle_action(2,0,0,1)
```

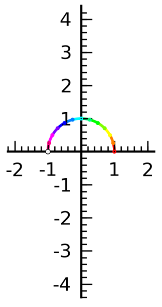


$\xmapsto{\begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}}$

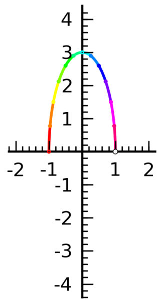


And here is the transformation that triples the y components and multiplies x components by -1. Note that it changes the orientation.

```
sage: p = plot_circle_action(-1,0,0,3)
```

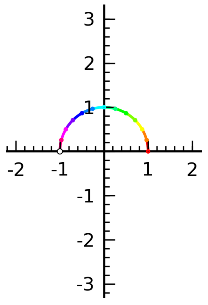


$\xmapsto{\begin{pmatrix} -1 & 0 \\ 0 & 3 \end{pmatrix}}$

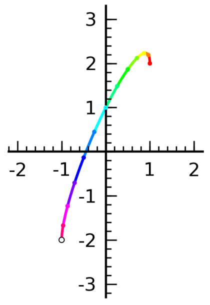


Here is the first skew transformation.

```
sage: p = plot_circle_action(1,0,2,1)
```

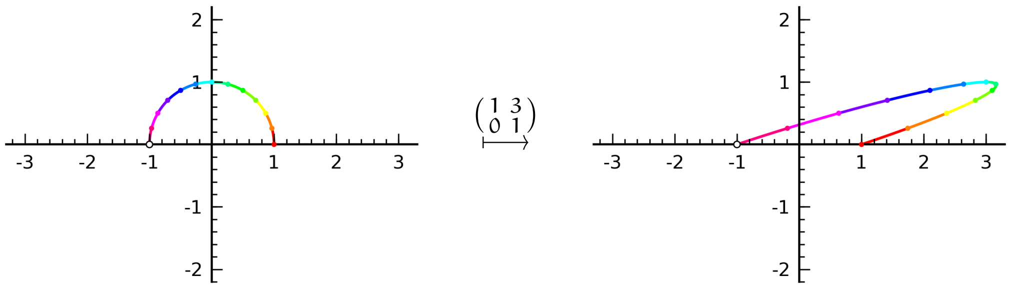


$\xmapsto{\begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}}$



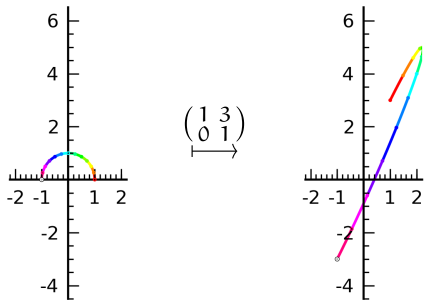
Here is the second skew transformation.

```
sage: p = plot_circle_action(1,3,0,1)
```



And here is the generic map.

```
sage: p = plot_circle_action(1,2,3,4)
```

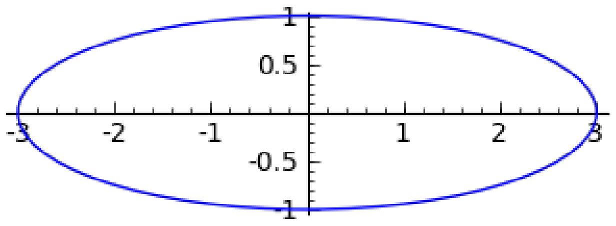


Circles map to ellipses

These pictures show the circles mapping to ellipses. Recall that an ellipse in $\mathbb{R}^2$ has a *major axis* axis, the longer one, and a *minor axis*.[1] We write σ_1 for half the length of the major axis, the distance from the center to the furthest-away point on the ellipse. We write σ_2 for the length of the semi-minor axis. These two axes are orthogonal.

```
sage: plot.options['axes_pad'] = 0.05
sage: plot.options['fontsize'] = 7
sage: plot.options['dpi'] = 500
sage: plot.options['aspect_ratio'] = 1
sage: sigma_1=3
sage: sigma_2=1
sage: E = ellipse((0,0), sigma_1, sigma_2)
sage: E.save("sageoutput/svd01.png", figsize=3.5)
```

[1] Ellipses have a couple of special cases. If the two axes have the same length then the ellipse is a circle. If an axis has length zero then the ellipse is a line segment.

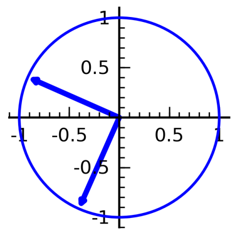


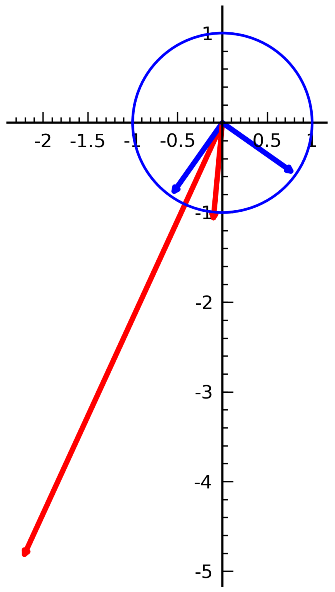
In general, under any linear map $t\colon \mathbb{R}^n \to \mathbb{R}^m$ the unit sphere maps to a hyperellipse. This is a version of the *Singular Value Decomposition* of matrices: for any linear map $t\colon \mathbb{R}^m \to \mathbb{R}^n$ there are bases $B = \langle \vec{\beta}_1, \ldots, \vec{\beta}_m \rangle$ for the domain and $D = \langle \vec{\delta}_1, \ldots, \vec{\delta}_n \rangle$ for the codomain such that $t(\vec{\beta}_i) = \sigma_i \vec{\delta}_i$.

Here is the example of the generic 2×2 matrix.

```
sage: plot.options['axes_pad'] = 0.05
sage: plot.options['fontsize'] = 7
sage: plot.options['dpi'] = 500
sage: plot.options['aspect_ratio'] = 1
sage: M = matrix(RDF, [[1, 2], [3, 4]])
sage: M.SVD()
(
[-0.404553584834 -0.914514295677]  [ 5.46498570422            0.0]
[-0.914514295677  0.404553584834], [           0.0 0.365966190626],

[-0.576048436766   0.81741556047]
[ -0.81741556047 -0.576048436766]
)
sage: beta_1 = vector(RR, [-0.404553584834, -0.914514295677])
sage: beta_2 = vector(RR, [-0.914514295677, 0.404553584834])
sage: delta_1 = vector(RR, [-0.576048436766, -0.81741556047])
sage: delta_2 = vector(RR, [0.81741556047, -0.576048436766])
sage: M*beta_1
(-2.23358217618800, -4.87171793721000)
sage: M*beta_2
(-0.105407126009000, -1.12532854769500)
sage: C = circle((0,0), 1)
sage: P = C + plot(beta_1) + plot(beta_2)
sage: P.save("sageoutput/svd02a.png", figsize=2)
sage: Q = C + plot(M*beta_1,color='red') + plot(delta_1)
sage: Q = Q + plot(M*beta_2,color='red') + plot(delta_2)
sage: Q.save("sageoutput/svd02b.png", figsize=4.5)
```





First, consider a point (x, y) on and ellipse. The ellipse is a conic section so x and y satisfy the relation $Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$ such that $B^2 - 4AC < 0$ for some parameters A, ..., F. (*?Describe major and minor axes, and say that if they cross at the origin then* $F = 0$*?*)

Consider a linear transformation $t\colon \mathbb{R}^2 \to \mathbb{R}^2$ and suppose that it is nonsingular (a singular transformation takes the domain space ellipse to a line segment or point, which we can consider a degenerate ellipse).

$$\mathrm{Rep}_{E_2,E_2}(t) = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

Then t has an inverse, so imagine that when the inverse is applied to (x, y) it gives $(\hat{x}, \hat{y})$.

$$\begin{pmatrix} \hat{x} & \hat{y} \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} x & y \end{pmatrix}$$

Substituting for x and y in the equation for a conic gives this formula about the inverse image.

$$A(a\hat{x} + c\hat{y})^2 + B(a\hat{x} + c\hat{y})(b\hat{x} + d\hat{y}) + C(b\hat{x} + d\hat{y})^2 + D(a\hat{x} + c\hat{y}) + E(b\hat{x} + d\hat{y}) + F = 0$$

Expand and gather terms to get this.

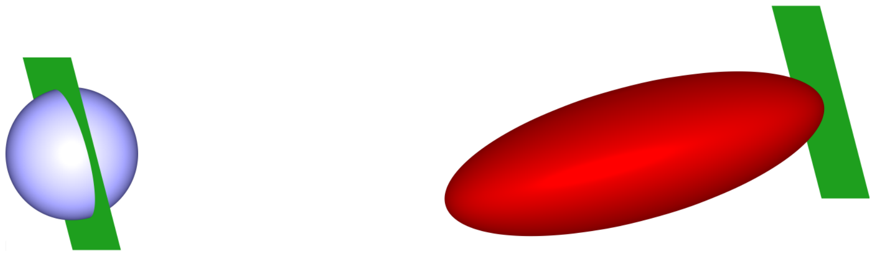
$$\begin{aligned} 0 = &(A \cdot a^2 + B \cdot ab + C \cdot b^2) \cdot \hat{x}^2 \\ &+ (A \cdot ac + B \cdot (ad + bc) + C \cdot bd) \cdot \hat{x}\hat{y} \\ &+ (A \cdot c^2 + B \cdot cd + C \cdot d^2) \cdot \hat{y}^2 \\ &+ (D \cdot a + E \cdot b) \cdot \hat{x} \\ &+ (D \cdot c + E \cdot d) \cdot \hat{y} \\ &+ F \end{aligned}$$

Thus the inverse of a conic is a conic.

The argument is here. Consider a vector

$$\vec{v} = \begin{pmatrix} x \\ y \end{pmatrix}$$

that lies on an ellipse. We can parametrize to get $x(t) = m\cos(t)$, $y(t) = n\sin(t)$ with $0 \leqslant t < 2\pi$.



Source of plot_action.sage

```
# plot_action.sage
# Show the action of a 2x2 matrix on the top half of a unit circle

DOT_SIZE = .02

def color_circle_list(a, b, c, d, colors):
    """Return list of graph instances for the action of a 2x2 matrix on
    half of the unit circle.  That circle is broken into chunks each
    colored a different color.
      a, b, c, d  reals  entries of the matrix
      colors  list of rgb tuples; len of this list is how many chunks
    """
    r = []
    t = var('t')
    n = len(colors)
    for i in range(n):
      color = colors[i]
        x(t) = a*cos(t)+b*sin(t)
        y(t) = c*cos(t)+d*sin(t)
        g = parametric_plot((x(t), y(t)),
                            (t, pi*i/n, pi*(i+1)/n),
                            color = color)
      r.append(g)
        r.append(circle((x(pi*i/n), y(pi*i/n)), DOT_SIZE, color=color))
    r.append(circle((x(pi), y(pi)), 2*DOT_SIZE, color='black',
                    fill = 'true'))
    r.append(circle((x(pi), y(pi)), DOT_SIZE, color='white',
                    fill = 'true'))
    return r

def plot_circle_action(a, b, c, d, n = 12, show_unit_circle = False):
    """Show the action of the matrix with entries a, b, c, d on half
    of the unit circle, as the circle and the output curve, broken into
    a number of colors.
     a, b, c, d  reals  Entries are upper left, ur, ll, lr.
     n = 12  positive integer  Number of colors.
    """
    colors = rainbow(n)
    G = Graphics()  # holds graph parts until they are to be shown
    if show_unit_circle:
        for f_part in color_circle_list(1,0,0,1,colors):
            G += f_part
    for g_part in color_circle_list(a,b,c,d,colors):
        G += g_part
    return plot(G)

THICKNESS = 1.75  # How thick to draw the curves
ZORDER = 5    # Draw the graph over the axes
def color_square_list(a, b, c, d, colors):
    """Return list of graph instances for the action of a 2x2 matrix
    on a unit square.  That square is broken into sides, each colored a
    different color.
      a, b, c, d  reals  entries of the matrix
      colors  list of rgb tuples; len of this list is at least four
    """
    r = []
    t = var('t')
    # Four sides, ccw around square from origin
    r.append(parametric_plot((a*t, b*t), (t, 0, 1),
                             color = colors[0], zorder=ZORDER,
                             thickness=THICKNESS))
    r.append(parametric_plot((a+c*t, b+d*t), (t, 0, 1),
                             color = colors[1], zorder=ZORDER,
                             thickness=THICKNESS))
    r.append(parametric_plot((a*(1-t)+c, b*(1-t)+d), (t, 0, 1),
                              color = colors[2], zorder=ZORDER,
                              thickness=THICKNESS))
    r.append(parametric_plot((c*(1-t), d*(1-t)), (t, 0, 1),
                             color = colors[3], zorder=ZORDER,
                             thickness=THICKNESS))
    # Dots make a cleaner join between edges
    r.append(circle((a, b), DOT_SIZE,
                    color = colors[0], zorder = 2*ZORDER,
                    thickness = THICKNESS*1.25, fill =  True))
    r.append(circle((a+c, b+d), DOT_SIZE,
                    color = colors[1], zorder = 2*ZORDER+1,
                    thickness = THICKNESS*1.25, fill =  True))
    r.append(circle((c, d), DOT_SIZE,
                    color = colors[2], zorder = ZORDER+1,
                    thickness = THICKNESS*1.25, fill =  True))
    r.append(circle((0, 0), DOT_SIZE,
                    color = colors[3], zorder = ZORDER+1,
                    thickness = THICKNESS*1.25, fill =  True))
    return r

def plot_square_action(a, b, c, d, show_unit_square = False):
    """Show the action of the matrix with entries a, b, c, d on half
    of the unit circle, as the circle and the output curve, broken into
    colors.
     a, b, c, d  reals  Entries are upper left, ur, ll, lr.
    """
    colors = ['red', 'orange', 'green', 'blue']
    G = Graphics()        # hold graph parts until they are to be shown
    if show_unit_square:
        for f_part in color_square_list(1,0,0,1,colors):
            G += f_part
    for g_part in color_square_list(a,b,c,d,colors):
        G += g_part
    p = plot(G)
    return p

plot.options['figsize'] = 2.5
plot.options['axes_pad'] = 0.05
plot.options['fontsize'] = 7
plot.options['dpi'] = 500
plot.options['aspect_ratio'] = 1
```

Bibliography

Robert A. Beezer. Sage for Linear Algebra. http://linear.ups.edu/download/fcla-2.22-sage-4.7.1-preview.pdf, 2011.

Jim Hefferon. Linear Algebra. http://joshua.smcvt.edu/linearalgebra, 2012.

David Joyner and William Stein. Open source mathematical software. *Notices of the AMS*, page 1279, November 2007.

Ron Brandt. *Powerful Learning*. Association for Supervision and Curriculum Development, 1998.

Sage Development Team. Sage tutorial 5.3. http://www.sagemath.org/pdf/SageTutorial.pdf, 2012a.

Sage Development Team. Sage reference manual 5.3. http://www.sagemath.org/pdf/reference.pdf, 2012b.

Shunryu Suzuki. *Zen Mind, Beginners Mind*. Shambhala, 2006.